INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2002; **55**:1439–1449 (DOI: 10.1002/nme.548)

All-hexahedral element meshing: automatic elimination of self-intersecting dual lines

Nestor A. Calvo*,† and Sergio R. Idelsohn

Centro Internacional de Métodos Computacionales en Ingeniería (*CIMEC*), *INTEC*, *Güemes 3450*, (*3000*) *Santa Fe*, *Argentina*

SUMMARY

There has been some degree of success in all-hexahedral meshing. Standard methods start with the object geometry defined by means of an all-quadrilateral mesh, followed by the use of the combinatorial dual to the mesh in order to define the internal connectivities among elements. For all of the known methods using the dual concept, it is necessary to first prevent or eliminate self-intersecting (SI) dual lines of the given quadrilateral mesh. The relevant features of SI lines are studied, giving a method to remove them, which avoids deforming the original geometry. Some examples of resulting meshes are shown where the current meshing method has been successfully applied. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: unstructured mesh; all-hexahedral; dual; self-intersections; 3D meshing

1. INTRODUCTION

Significant advance has been made in the area of all-hexahedral mesh generation in the last few years [1]. All the surviving general-purpose methods deal (directly or not) with the combinatorial dual of the mesh [2–8]. The meaning of the dual and the methods based on it can be found in any of these references.

These methods start from an all-quadrilateral mesh as input; this mesh is used to define the boundary geometry. In this work, this boundary surface is further required to have the topology of a two-dimensional sphere, that is a single closed surface without holes.

There are, however, at least three sources of trouble in developing a fully automated generator in such a way:

(1) Some dual lines to the given boundary mesh are self-intersecting (SI), they are present in almost every well-shaped unstructured mesh. Each boundary dual line bounds a surface (to be made) dual to a sheet of hexahedra. If the line is SI, the surface must

*Correspondence to: Nestor A. Calvo, Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC), INTEC, Güemes 3450, (3000) Santa Fe, Argentina
†E-mail: ncalvo@ceride.gov.ar

Received 26 March 2001
Revised 16 January 2002
Accepted 20 February 2002

be SI too, and they are very difficult to deal with. The exact source of problems with SI lines or surfaces is approach dependent, but all the main methods acknowledge these difficulties [5, 6, 8]. This problem is still one of the main sources of failure when meshing.

(2) Topological monsters appear because the generation stage deals only with the topology of the problem. The term refers to objects agreeing with the topological requirements but giving invalid hexahedra in order to be used in the finite element method. Mostly they give hexahedra sharing more than a face, edge or vertex with another element and even sharing some sub-entities with itself. Some problems of this kind have already been solved by different approaches with varying degree of success [5, 8–11]. There is not yet a fully satisfactory method to remove them.

(3) Smoothing algorithms have extremely low speed, at least those successful in disentangling meshes and improving their qualities. Although faster algorithms have been implemented, so far they have had little success in reverting elements with a negative Jacobian at some vertices [11–13].

This work deals only with the dual of the external mesh and the first problem mentioned before. As there is not yet any successful algorithm to handle SI surfaces, there is presented an algorithm to automatically modify the boundary mesh in order to eliminate the SI lines, thus no SI surfaces will be made.

There are already in the literature some methods for the elimination of SI's from the boundary dual [5, 6, 14]. In particular, the work of Folwell and Mitchell [5] presents some of the basic features of the process needed to remove SI lines by collapsing quadrilaterals.

This work is intended to present a systematic framework for the study of the topological properties of SI lines and to discuss some available methods for the removal of self-intersections. The goals are to avoid modifications on the input geometry and to minimize the quality loss in the input mesh.

The author's previous implementation [6] is reviewed and the results on some testing input meshes are shown, in order to see the effect of the SI removal routine for problems that were successfully meshed with hexahedra without topological monsters.

2. SELF-INTERSECTING CLOSED LINES

2.1. Definitions and notation

We will analyse planar closed curves with the further requirements that they are not self-tangent and that they have no triple self-intersection points. These are also known as unicursal curves. We will also establish a method for traversing the line, so it can be considered as a directed line. This ordering will be a means to make deductions in an easy way but, otherwise, it is unnecessary.

Figure 1 shows a SI closed line with a two-number labelling at each node. This labelling is the order in which each cross must be passed over when the line is traversed. There are n nodes connected by $2n$ curve segments. At any node such as $(6, 19)$ there are four incident segments: two incoming segments like 5–6 and 18–19 and two outgoing ones as 6–7 and 19–20.

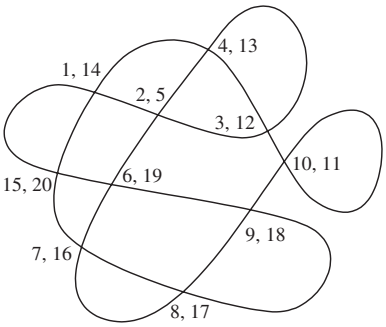

Figure 1. Node labelling a self-intersecting line.

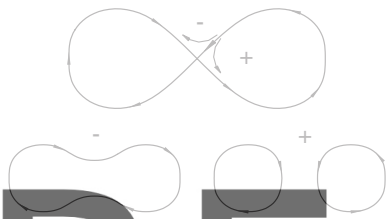

Figure 2. Positive and negative split of a self-intersection.

We will make extensive use of the Jordan curve theorem, stating that a smooth (differentiable, non-zero tangent vector) simple (non-SI) closed line divides the plane into two disjoint regions each of which is homeomorphic to a disk. This theorem seems obvious on the plane and can be shown using Green's theorem. It does not hold on surfaces with holes. For such surfaces, the spherical and planar problems are essentially the same, however almost nothing will be valid in surfaces with genus (number of holes) greater than zero.

2.2. *Splitting self-intersecting lines*

Each self-intersection of a curve can be spliced in two ways. Let us label positive a split joining segments coming to the node with segments leaving the node, and negative to the opposite case. Figure 2 shows the labelling scheme.

A positive split always leaves two curves (may be non-disjoint) and preserves the original direction of the segments. On the other hand, by negatively splitting, one branch must be inverted, thus leaving a single curve.

Splitting all the n nodes gives $2n$ possible outcomes. Each one is a set of disjoint closed lines; some sets being made of a single line. When each split leaves a connected line set, the outcome is a single line.

Two lines are obtained by making a positive split like the one at node $(6, 19)$ in Figures 3 and 4. In this case, both of them having common points such as $(15, 20)$ as well as self-intersections like $(9, 18)$.

One consequence from the Jordan theorem is that each node label has one even and one odd number. Consider the split made at Figure 4, a round trip starting from node 6 meets

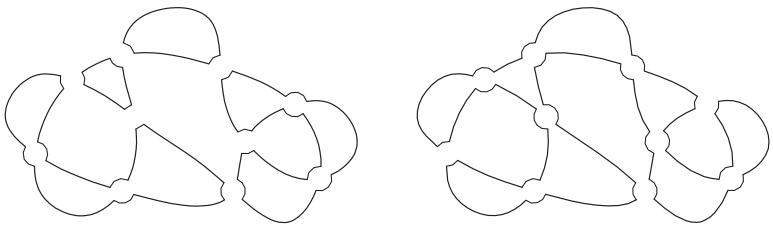

Figure 3. Two different outcomes resulting from splitting all the nodes.

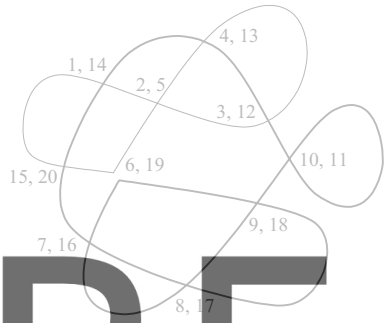

Figure 4. Splitting the line into two.

the other closed circuit an even number of times, and also each self-intersection of the trip is found twice.

2.3. *Circular diagram and splitting*

For computational analysis and manipulation, we made a simple representation of all the relevant structure of SI lines. The sequence of crossings found when travelling the line is mapped on a circumference as seen on Figure 5. The crossings belonging to the same node are identified by means of inner lines joining them.

At the computer, this information is allocated in a circular one-dimensional array of integers in which each element (crossing) is pointing to the other crossing of the same node.

Let us see the effect of both types of splitting by means of diagrams.

In the upper diagram of Figure 6 there are two arcs (2–9 and 11–20) with closing chords, representing the two pieces resulting from a positive split at node (1, 10). These pieces are mutually intersected at nodes such as (4, 15) whose lines in the diagram cut the splitting line. Self-intersections of the pieces remain as lines (as 13–16) which do not cross (are parallel to) the splitting line.

A negative split reverses one of the pieces and attaches it to the other, resulting in a single line with one less self-intersection. Note the reversal of the traversing order in one of the arcs.

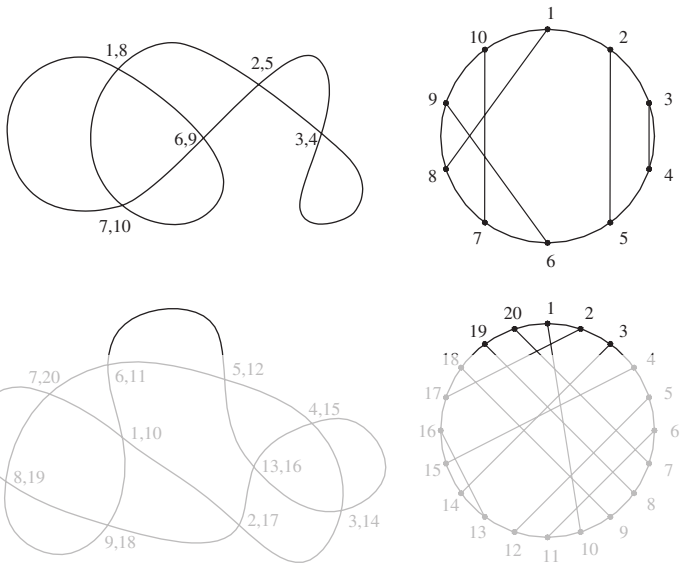

Figure 5. Circular diagrams.

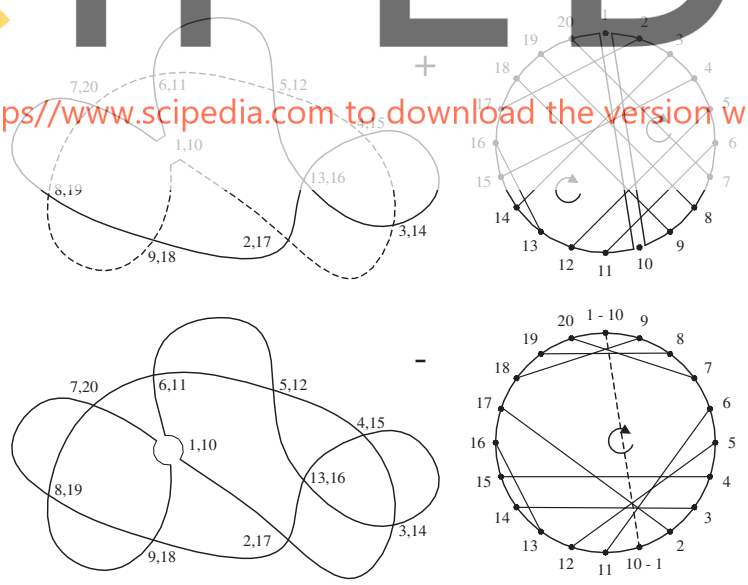

Figure 6. Circular diagrams for a positive and a negative split at a node.

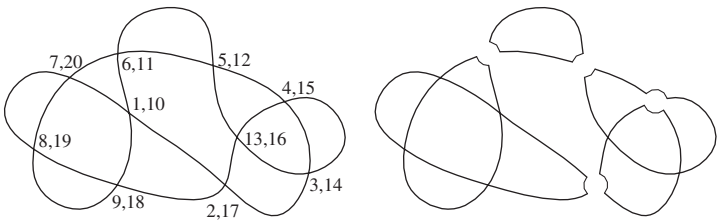

Figure 7. Splitting to avoid self-intersections.

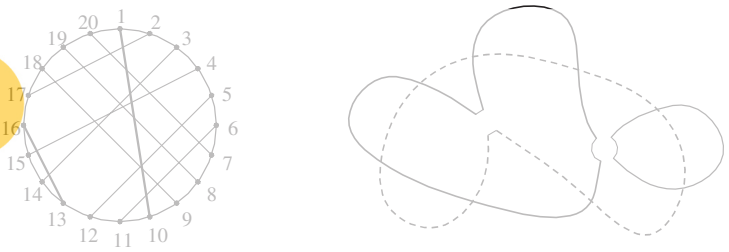

Figure 8. Optimal splitting.

2.4. *Minimal splitting*

In order to avoid self-intersections, it is not necessary to split the line at every node. Splitting at a few nodes instead, as Figure 7 shows, results in a set of simple lines, which is enough for meshing purposes.

A negative split reduces the number of self-intersections by only one, whereas a positive split leaves two closed lines with some of the previous self-intersections as mutual intersections. Therefore, we will use only positive-type splitting.

Mutual intersections were already identified in the diagram with the lines crossing the splitting line. Because lines $(1,10)$ and $(13,16)$ in Figure 8 intercept all the other lines, only two splits are enough in order to obtain simple lines.

The question is how to make an algorithm to find such minimal set of lines in any problem.

One relevant datum in the search of that minimal set is the number of lines intercepted by each line in the circular diagram. In Figure 8, the line corresponding to node $(1,10)$ intercepts eight other lines. We will refer to that number as the 'cutting index' of the node $(1,10)$.

Any line divides the circular diagram into two arcs, the cutting index is the number of unpaired crossings in each of these arcs, i.e. the number of crossings in the arc minus two crossings for each parallel line whose ends are in the arc. Because the circular diagram have $2n$ crossings for n nodes and, as was stated, each line connect one even and one odd crossing number, each arc have an even number of crossings, thus all cutting indices are even numbers.

With the data-structure above mentioned for the diagram, the cutting indices are easily found. Nevertheless, finding the minimal set of lines cutting the rest of the lines is not so simple. One possible solution seems to be the recursive use of a maximal line i.e. one with

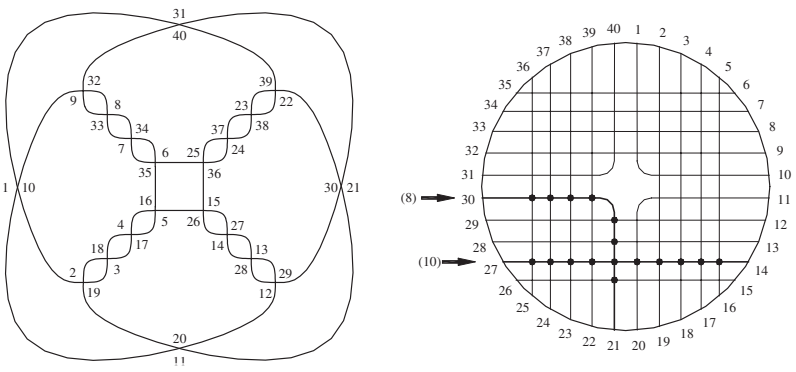

Figure 9. Counterexample for the hypothesis that recursive use of maximal lines gives the optimal cutting set.

maximal cutting index. We use such quick and simple approach, but we know that is not the best way for every case. Figure 9 shows a counter example.

Four bent lines like (21,30) cut eight other lines each, whereas all the straight lines intercept ten lines. Six splits are needed by using any of the maximal lines because this implies further use of all the parallel remnants. The four minimal lines instead, make the optimal cutting set for this case.

3. THE SPLITTING ON THE MESH SIDE

Removing a node is the dual operation to removing an element of the input mesh.

An element corresponding to a self-intersection is schematized at the centre of Figure 10(a), whereas in 10(b) the effect of removing such element is shown. Such process (as used by Hannemann [6]) have visible bad consequences if any of the edges of the removed element were part of a geometric edge, i.e. an edge of the object being meshed. Figure 10 shows a geometric edge in 10(a′) whereas 10(b′) shows the effect of removing an element.

In order to avoid such deformations, we made [14] (some years ago) a suitable subdivision of the element as shown in Figure 10(c) and the enlarged detail 10(c′). The changes were made inside the original element without any changes on any of the original edges. The outcome, at the dual side, is made up of two lines crossing twice inside the element and a little ‘circular’ new line. This last line is needed in order to avoid generating a digon (two-edged region dual to a pair of quads sharing two adjacent edges[‡]).

Because of the resulting low quality, such an approach was abandoned in favor of other means to resolve the self-intersections problem (into the inner 3D mesh). These methods are still wished for but unsuccessful so far. So we returned to this scheme but with another method.

[‡]This is the simplest of the topological monsters mentioned in the introduction.

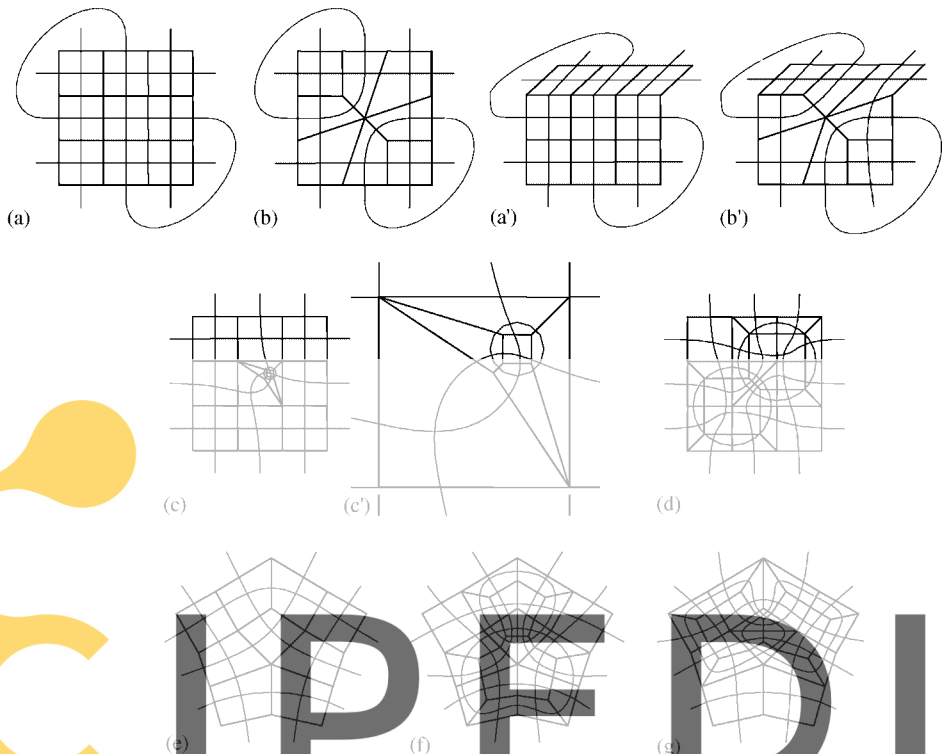

Figure 10. Effect of splitting on the mesh and some ways for removing an element.

In Figure 10(d), the currently implemented method is shown. The lines no longer intercept and there are two new lines avoiding digons, which span neighbouring elements. The elements so created have qualities that are far more acceptable, even when the neighbourhood of the element is not orthogonal, as shown in 10(e), 10(f) and 10(g) in the same figure.

The method followed by Folwell and Mitchell [5] requires a selection process to be made, they 'pillow' (add one of those little circular lines) the SI node whenever they must avoid modifying a geometrical edge, and then they pillow again every digon obtained. The outcome is very similar to ours.

As can easily be shown, many methods can be used. The goals are to obtain a method simple enough to be automatically used in every situation, to avoid deformations on the input geometry and to minimize the quality loss of the input mesh in order to avoid a time expensive second smoothing stage. There is not such a thing as 'the better method'. With our method, the original edges are not modified, and the process is fully automated.

4. EXAMPLES

As previously mentioned, we show here some meaningful boundary meshes for which, before the SI removal, we were not able to make an all-hexahedral mesh. Now we made such

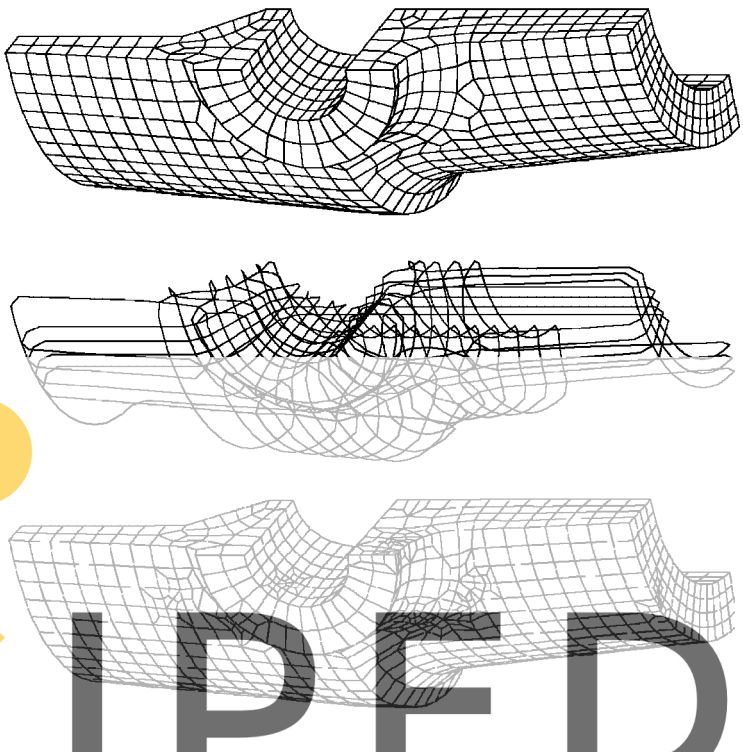

Figure 11. Tube fitting, original mesh, single self-intersecting line and resulting mesh.

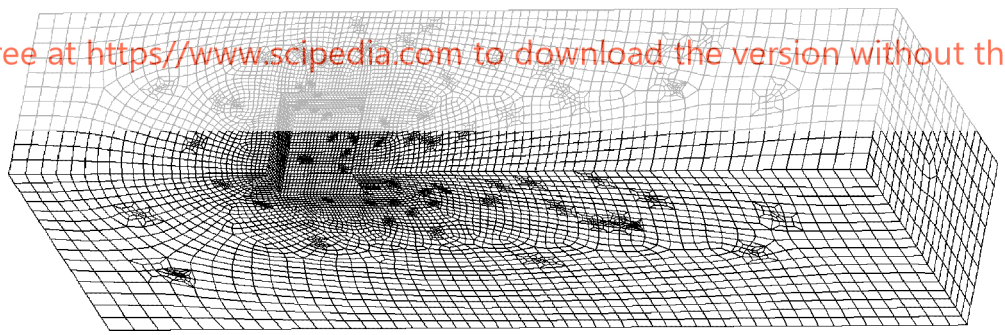

Figure 12. Mesh for a rectangular channel with a cubic obstruction.

meshes without any other problem appearing. There were some meshes (not presented here) were other topological problems remains and thus we are not yet capable to mesh inside them.

Figure 11 shows a quad mesh for an oil tube fitting. The upper mesh is the original one. In order to show the magnitude of the problem we extracted from this mesh a single dual line intercepting itself many times. At the bottom of the same figure, the resulting automatically cleaned mesh is shown.

Similar results are shown in Figures 12 and 13.

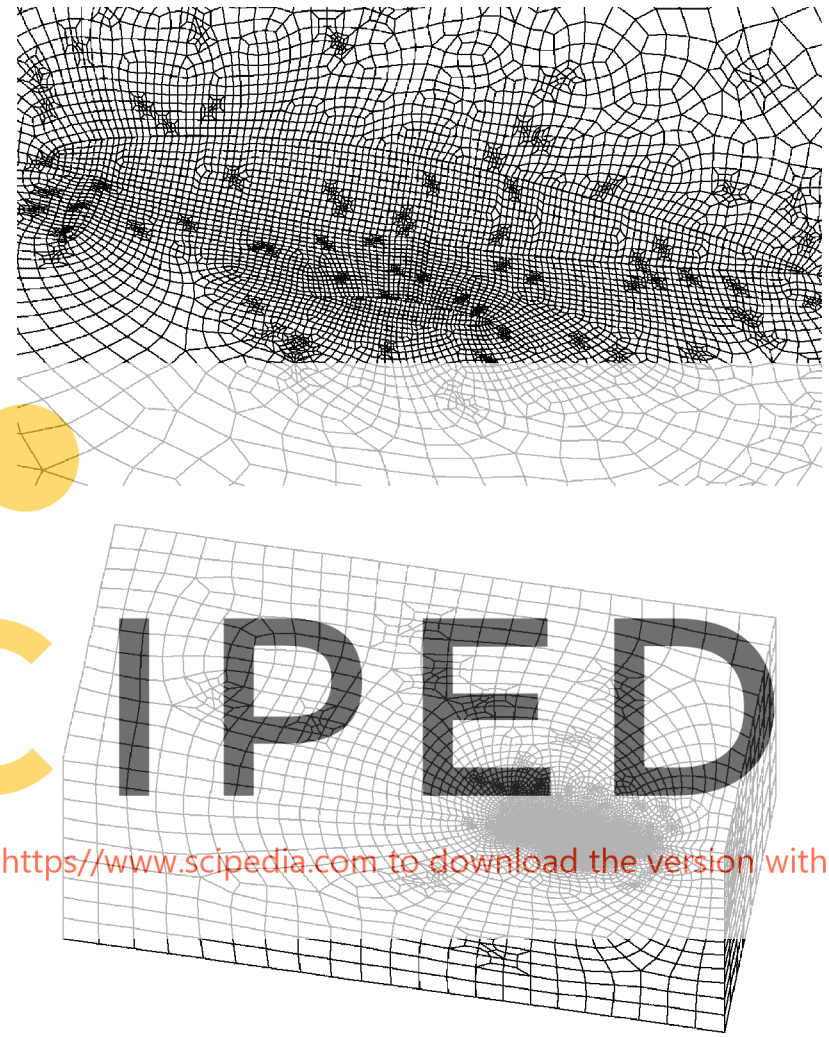

Figure 13. Effect of removing self-intersections in a mesh for the water around a sailboat (detail and complete mesh).

5. CONCLUSIONS AND FURTHER WORK

Before the implementation of automatic elimination of self-intersections, we were only able to mesh inside a few external meshes, mostly made *ad hoc*. Now the proportion of success/failure has been reversed, with a remainder of a few meshes where the problem is not related to the external self-intersections. The source of misconnection among elements relies on a certain kind of topological monsters, which are not successfully removed with our current algorithms. However, we were not able to realize that such problems exist at all, until we avoided self-intersections in order to fix one problem at a time.

There are two negative aspects of this approach deserving consideration.

One is the fact that the cutting set giving the minimum number of splits is by no means the geometrical optimum. This is because there are some splits made next to previous ones, lowering even more the quality of the elements involved. This problem can be seen in the examples shown before. The true optimum is the minimal set of splits with enough separation among them; or better still, the splitting set minimizing the quality loss. This result is not implemented (if it could be accomplished at all) but the current results are enough for our current purposes.

The other (far more important) problem arises because sometimes one needs to mesh by pieces. Being unable to predict *a priori* what changes the program will make on any surface, makes it impossible to match two surfaces in order to paste the pieces together (giving a conformal mesh). This problem also applies when the original geometry is not 'spherical'. When the body has holes, the common practice is to make a cut along any 'handle' adding the same mesh on both sides of the cut.

The approach introduced here has proven to be powerful experimental help and an adequate emergency tool for meshing, but we acknowledge that this is not the best solution although it is the best current solution.

Further work is oriented towards the internal treatment of self-intersections and pathological connectivity. We hope (we have clues to hope) that succeeding in repairing badly connected elements will allow the previously unsuccessful attempts to resolve the self-intersections internally.

REFERENCES

1. Owen SJ. A survey of unstructured mesh generation technology, quad/hexahedral meshing. <http://www.andrew.cmu.edu/user/sowen/survey/hexsurv.html>
2. Price MA, Armstrong CG. Hexahedral mesh generation by medial surface subdivision: part I. Solids with convex edges. *International Journal for Numerical Methods in Engineering* 1995; **38**:3335–3359.
3. Price MA, Armstrong CG. Hexahedral mesh generation by medial surface subdivision: part II. Solids with flat and concave edges. *International Journal for Numerical Methods in Engineering* 1997; **40**:111–136.
4. Tautges T, Blacker T, Mitchell S. The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering* 1996; **39**:3327–3349.
5. Folwell NT, Mitchell SA. Reliable whisker weaving via curve contraction. *Proceedings of the 7th International Meshing Roundtable*, 26–28 October 1998, Dearborn, Michigan, U.S.A.
6. Müller-Hannemann M. Hexahedral mesh generation by successive dual cycle elimination. *Proceedings of the 7th International Meshing Roundtable*, 26–28 October 1998, Dearborn, Michigan, U.S.A.
7. Kober C, Müller-Hannemann M. Hexahedral mesh generation for the simulation of the human mandible. *Electronic Proceedings of the 9th International Meshing Roundtable*, New Orleans, LA, U.S.A., 2–5 October 2000. <http://www.imr.sandia.gov/9imr.html>
8. Calvo N, Idelsohn S. All-hexahedral element meshing: generation of the dual mesh by recurrent subdivision. *Computer Methods in Applied Mechanics and Engineering* 2000; **182**:371–378.
9. Tautges TJ, Mitchell SA. Whisker weaving: invalid connectivity resolution and primal construction algorithm. *Proceedings of the 4th International Meshing Roundtable*, Sandia National Laboratories, October 1995.
10. Mitchell SA, Tautges TJ. Pillowing doublets: refining a mesh to ensure that faces share at most one edge. *Proceedings of the 4th International Meshing Roundtable*, Sandia National Laboratories, October 1995.
11. Knupp PM. Hexahedral mesh untangling & algebraic mesh quality metrics. *Electronic Proceedings of the 9th International Meshing Roundtable*, New Orleans, LA, U.S.A., 2–5 October 2000. <http://www.imr.sandia.gov/9imr.html>
12. Knupp PM. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Parts I and II. *Technical Report SAND 99-0709J*, Sandia National Laboratories, 1999.
13. Calvo NA, Idelsohn SR. All-hexahedral mesh smoothing with a node-based measure of quality. *International Journal for Numerical Methods in Engineering* 2001; **50**:1957–1967.
14. Calvo N, Idelsohn S. Generador automático de mallas de hexaedros: presentación del método y avances en la implementación. *Mecánica Computacional Vol. XVI*, Asociación Argentina de Mecánica Computacional, 1996.

